Quality engineering process for the Program Design Phase of a generic software life cycle

Witold Suryn[1], Abdelilah Kahlaoui[2], Elli Georgiadou[3]

[1]Software and Information Technology Engineering Dept, École de technologie supérieure. 1100 Notre Dame St. West, Montréal, Québec, H3C 1H3 Canada. E-mail: witold.suryn@etsmtl.ca

[2]Master student. Software and Information Technology Engineering Dept, École de technologie supérieure. 1100 Notre Dame St. West, Montréal, Québec, H3C 1H3 Canada. E-mail: akahlaoui@hotmail.com

[3]Middlesex University
School of Computing Science
The Burroughs, Hendon, London NW4 4BT, UK
e.georgiadou@mdx.ac.uk

Abstract

This paper presents the design of a quality engineering process applicable in the program design phase of a generic software life cycle. The presented process model aims to guide the software quality engineer from the beginning of the phase to its completion defining the collaboration that should take place between the program designer and the software quality engineer. The paper also discusses the integration of the software quality engineering process with the software development process.

1. Introduction

Developers often see quality as something necessary and desired but quality is not always on the top of their priority list. As they rush through the project, quality-related activities are being set aside in order to develop and deliver the software. Sometimes they may even feel it's a time consuming process with rather limited pay-offs. The reality is that quality is not considered a luxury anymore, and its negligence is likely to become one of the most expensive mistakes. As times change and customers become more and more demanding, an increasing group of leading companies successfully use quality concepts to drive their key products and win the market.

The pivotal concept presented in this paper employs the idea that to obtain high quality of software the software development life cycle has to be supported by and integrated with the appropriate software quality engineering process (SQEP) as a natural part of it. Such a process should also be methodology-independent and adjustable to the culture and practices of an organisation that chooses to apply it.

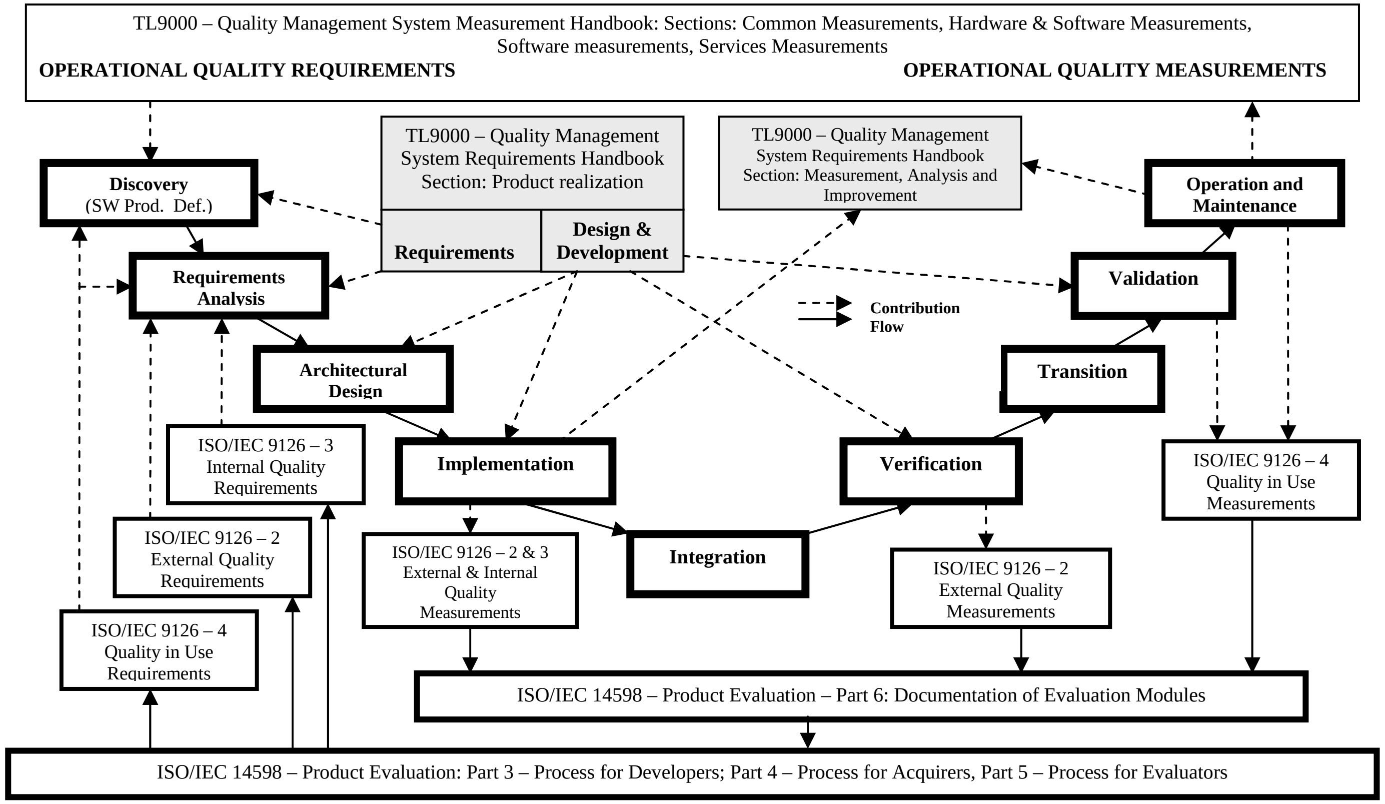
2. Objectives

The integrated Suryn-Abran model [1] maps the activities of a software development generic life cycle to recognised standards for software quality (Figure **1**). As the model shows, most phases are supported by relevant software quality international standards to the extent of specialised quality measures and dedicated quality models and processes. However, the phase of Conceptual Design which encapsulates two rather critical components namely *System Design* and *Program Design* remain unaddressed.

This paper proposes the processes and models helping to fill this gap with the particular support for the Program Design Phase (PDP). The methodology applied during this research consists of three steps:

1. Analysis of ISO/IEC 9126 [2, 3] in order to identify quality characteristics and subcharacteristic that could be applied to the PDP;
2. Elaboration of a program design process applying the IEEE 1016 standard [4];
3. Creation of a Quality Program Design Process (QPDP) than can be integrated and performed in parallel with the ordinary PDP [5, 6, 7, 8, 9].

The above methodology was also successful in identifying quality measures that could be effectively applied during the Program Design Phase.


TL9000 – Quality Management System Measurement Handbook: Sections: Common Measurements, Hardware & Software Measurements, Software measurements, Services Measurements
OPERATIONAL QUALITY REQUIREMENTS
OPERATIONAL QUALITY MEASUREMENTS
TL9000 – Quality Management System Requirements Handbook Section: Product realization
Requirements
Design & Development
TL9000 – Quality Management System Requirements Handbook Section: Measurement, Analysis and Improvement
Discovery (SW Prod. Def.)
Requirements Analysis
Architectural Design
Implementation
Integration
Verification
Transition
Validation
Operation and Maintenance
Contribution
Flow
ISO/IEC 9126 – 3 Internal Quality Requirements
ISO/IEC 9126 – 2 External Quality Requirements
ISO/IEC 9126 – 4 Quality in Use Requirements
ISO/IEC 9126 – 2 & 3 External & Internal Quality Measurements
ISO/IEC 9126 – 2 External Quality Measurements
ISO/IEC 9126 – 4 Quality in Use Measurements
ISO/IEC 14598 – Product Evaluation – Part 6: Documentation of Evaluation Modules
ISO/IEC 14598 – Product Evaluation: Part 3 – Process for Developers; Part 4 – Process for Acquirers, Part 5 – Process for Evaluators
Figure 1: Integrated model by Suryn and Abran

3. Measures and measurement

ISO/IEC 9126 is the ISO standard that addresses software quality characteristics and subcharacteristic [2, 3]. It also gives a set of internal and external measures to validate quality of software. These measures are product-oriented (i.e. pertinent to a software product) and as such, not always well suited for phases where no code is produced. Our first task was to extract the quality attributes and measures that can be applied to the PDP. We recognise that significant work is needed in order to adjust existing measures, and propose measurements that satisfy the most critical phases in software development, especially architecture and program design phases.

4. Designing for quality

The following sections describe the QPDP. This process adds a set of quality activities to the classical activities of a program design, in order to obtain a program design that satisfies the software quality requirements. The reader will notice that this process is performed in parallel with the usual software development process. This in turn makes the Software Quality Engineer (SQE) a member of the software development team and not just an inspector taking action only when the development is completed. The SQE is there to give support and guide the designers to achieve their objectives of quality. As the designer becomes more comfortable with quality concepts, that guidance might be less needed, but the SQE remains proactive nonetheless.

4.1. Classical Program design

This section presents a program design process inspired by the IEEE-1016 standard [4] and approaches discussed in [10, 11, 12]. The process is analysed in a step-by-step manner in order to show how proposed QPDP can be combined with it.

4.1.1. Program design elements

The below definitions are based on these used in IEEE 1016 [4].

Internal Entity: a definition of a set of attributes and operations that work together to produce the entity behaviour. An entity has a purpose that justifies why it exists and a function that sets what the entity does.

Design entity is an element (component) of a design that is structurally and functionally distinct from other elements and that is separately named and referenced. The design entity is associated to other entities through *Relationships*. This way an entity can use services of other entities the same way it can be used by other entities.

Data Entity: a set of information used by the entity to accomplish its *function.*

Operation Entity : defines an entity's behaviour. Operation behaviours form the entity's function

Entity Relationships: a link between two entities. A relationship between two entities means that a design entity uses, accesses or just knows about the related design entity.

Design interface: defines the services offered by a design entity.

4.1.2. Program design process

In this section we will define the activities considered as required for a designer to perform in order to produce a software program design. These activities are:

Internal Entity Design: The goal of this activity is to identify all possible entities in the software program.

Entity Relationships Design: In this activity, the designer should define the relationships that tie the entities together.

General Operations Definition: In this activity, the designer should define the general behaviour of each entity operation. The designer should also specify the data needed by the operations to do its work and the data it produces.

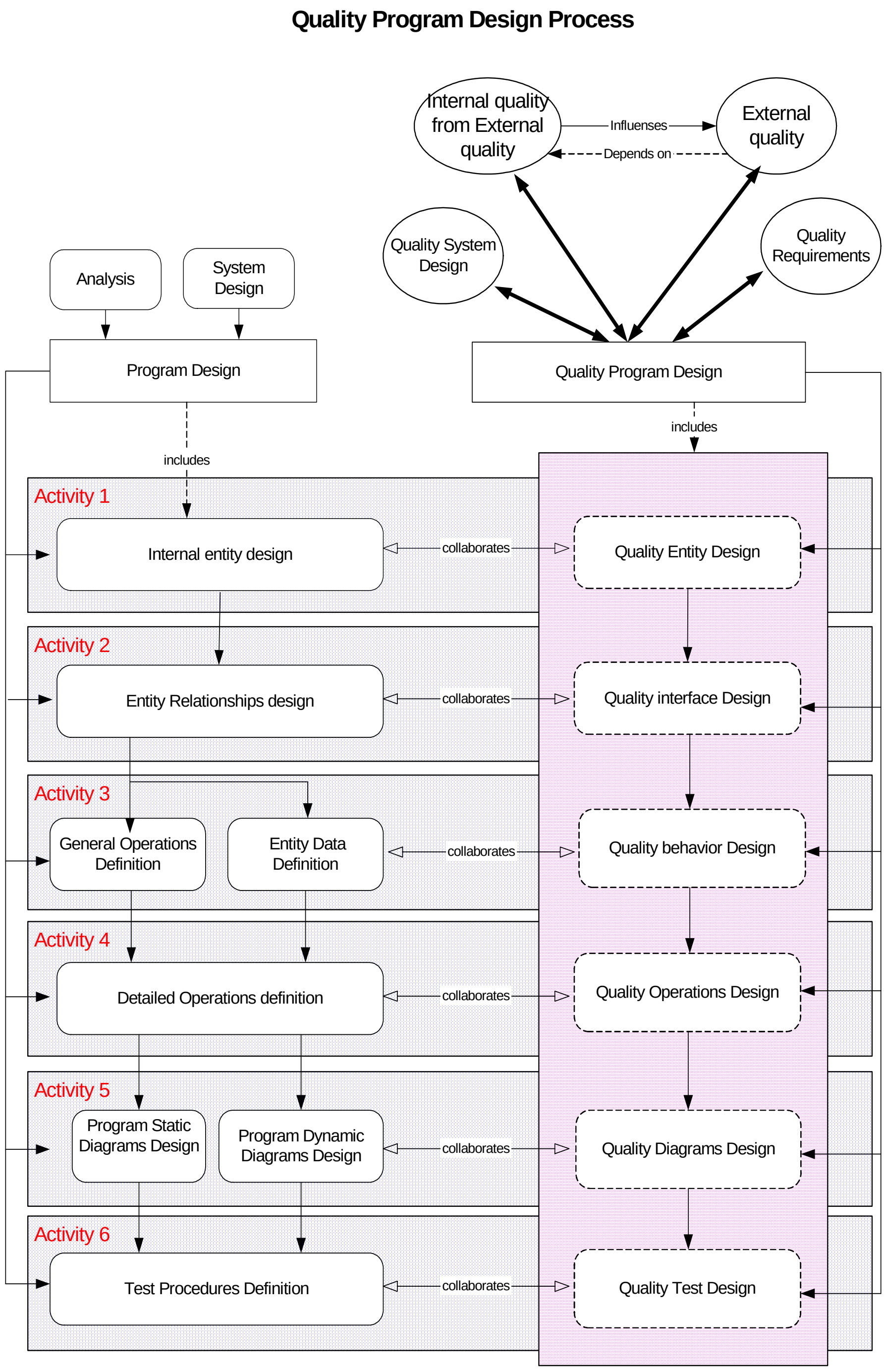


Figure 2: Program design quality engineering process

Entity Data Definition: In this activity, the designer should define the data used by the entity to accomplish its function. These data specify the state of an entity and are available for all operations defined in the entity.

Detailed Operations Definition: This activity allows the designer to add any pertinent information (e.g.: constraints, data validation rules, pseudocode etc.) related to the entity operations behaviour in order to help the programmer implement it.

Program Static Diagrams Design: In this activity, the designer should set up all diagrams that describe the structure of the software program.

Program Dynamic Diagrams Design: In this activity, the designer should set up all diagrams that describe the behaviour of the software program during its execution.

Test Procedures Definition: The goal of this activity is to define procedures to test the software program.

4.2. Quality Program Design Process

This section explains the activities that should be performed by a SQE in order to make sure that the quality requirements are met during the program design process. The quality program design process is executed in parallel with the program design. Each activity of the classical program design is mapped to a corresponding quality activity responsible for providing the required quality engineering-related information. These activities are:

Quality Entity Design: in this activity, the quality engineer should specify the quality characteristics to respect during the entity design activity. He identifies also the applicable and measurable quality attributes for entity design.

Quality Interface Design: in this activity, the quality engineer should specify the characteristics to respect during the interface design activity. He identifies also the applicable and measurable quality attributes for interface design.

Quality Behaviour Design: in this activity, the quality engineer should specify the characteristics to respect during the behaviour design activity. The SQE also identifies the applicable and measurable quality attributes for behaviour design.

Quality Operation Design: in this activity, the quality engineer should specify the characteristics to respect during the operation design activity. He identifies also the applicable and measurable quality attributes for operation design.

Quality Diagrams Design: in this activity, the quality engineer should specify the characteristics to respect during the diagrams design activity. The SQE also identifies the applicable and measurable quality attributes for diagrams design.

Quality Test Design: in this activity, the quality engineer should specify the characteristics to respect during the test design activity. The SQE also identifies the applicable and measurable quality attributes for test design.

4.3. Quality Program Design Artefacts

The following list is given as an example of the artefacts that can be generated in the PDP. It can be used as it is or adjusted according to different organisation needs.

Entity Design document: this document is produced by the software designer. It should include the definition of all entities, data, operations as well as the representation of the relationships between entities.

Realisation document: this document is produced by the software designer. It should include the static and dynamic diagrams developed in activity #5. It should also include any pertinent information that can be helpful for the programmer in the implementation phase.

Test case document: this document is produced by the software tester or designer. It should include a definition of the test procedures, the entities to be tested and the cases that should be covered by the test.

Quality Program Design document (QPDD): this is a document produced by the software quality engineer. It could be split into several subdocuments. It should include all the quality attributes to take into consideration and the recommended practice to satisfy the quality requirements. The following section shows how to elaborate a QPDD.

5. Quality Program Design Document

This section describes the process that should be followed by the SQE in order to produce the Quality Program Design Document (QPDD). There are 7 steps (Fig.3) in this process:

Quality program design attributes definition

In this step, the SQE should define all applicable quality program design attributes. The inputs for this step are:

- Internal quality characteristics and subcharacteristics from ISO/IEC 9126 standard
- Quality system design
- Present document

Prioritising quality attributes

In this step, the SQE should prioritise quality program design attributes defined in step 1. The criteria used to prioritise these attributes may vary from one organisation to another however both the size and the type of the project could be considered as a starting point.

Measures identification

In this step, the SQE should set up all applicable measures. Based on the present document, the SQE should be able to identify all measures related to the quality attributes defined in step 1.

Target measurement values specification

In this step, the SQE should specify the acceptable values for the measures identified in step 3. These values may vary depending on the project type and size.

Recommended practice elaboration

In this step, the SQE may adapt (combine activities together or add its own activities) the activities proposed in the quality program design process in order to respect the organisation internal standards.

Engineering process adaptation

In this step, the SQE should provide the necessary guidelines to help the designer satisfy the quality attributes defined in step1.

Design artefacts specification

Based on the project type and size and the organisation's internal standards, the SQE should define the pertinent artefacts to produce by the designer.

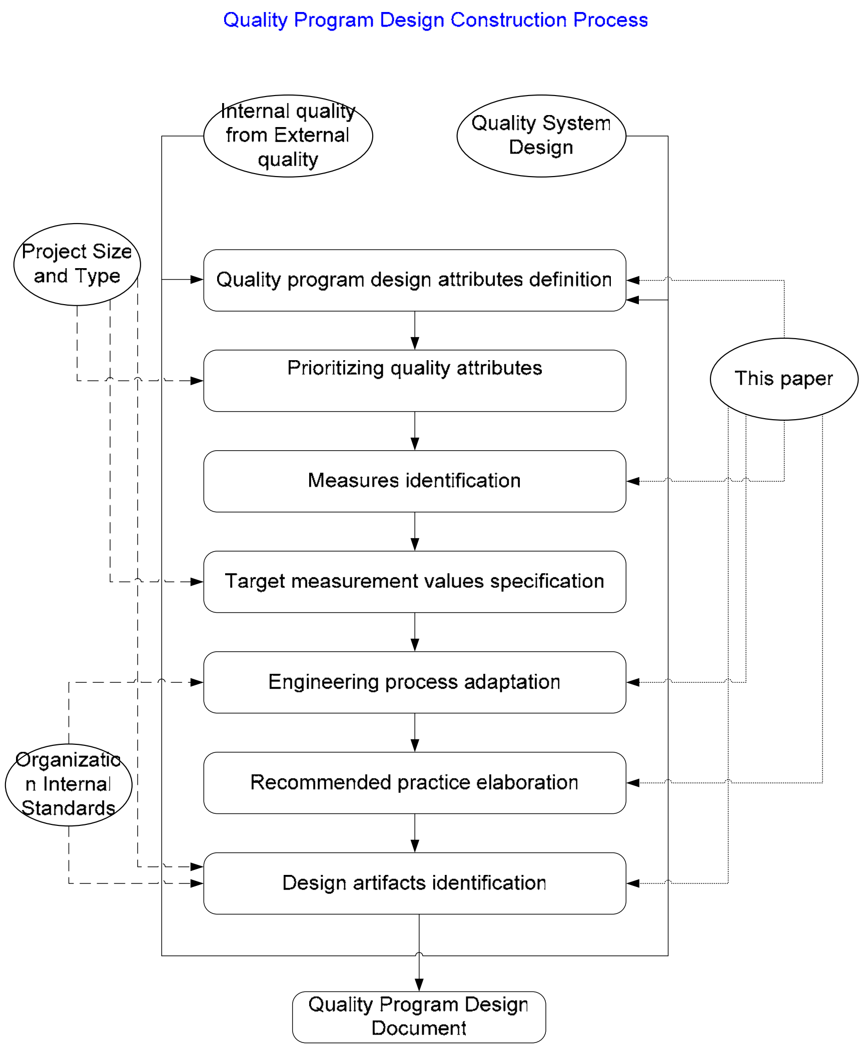

Quality Program Design Construction Process
Internal quality from External quality
Quality System Design
Project Size and Type
Quality program design attributes definition
Prioritizing quality attributes
Measures identification
Target measurement values specification
Engineering process adaptation
Recommended practice elaboration
Design artifacts identification
This paper
Organization Internal Standards
Quality Program Design Document

Figure 3: Quality program design document construction process

6. Collaboration Process

This section describes the process of collaboration between the SQE and the designer during an activity of program design (Fig.4). The same process is repeated for each activity; only the scope changes.

Stabilizing Quality Attributes

The SQE and the designer should work together in order to standardise and reach an agreement on quality attributes specified by the SQE in the QPDD.

Standardised Measures

The SQE and the designer should work together in order to standardise and reach an agreement on the measures identified by the SQE in his QPDD.

Design

The designer should build the program design following the guidelines given by the SQE in the QPDD.

Design Documentation

The designer should document the program design.

Design Documentation Inspection

The SQE should inspect the program design document produced in phase 4.

Design Review

The SQE should participate in the review the program design produced in phase 3.

Measuring design

The SQE should perform the measurements agreed in phase 2.

Acceptation

The SQE should inspect and validate the quality attributes agreed in phase 1.

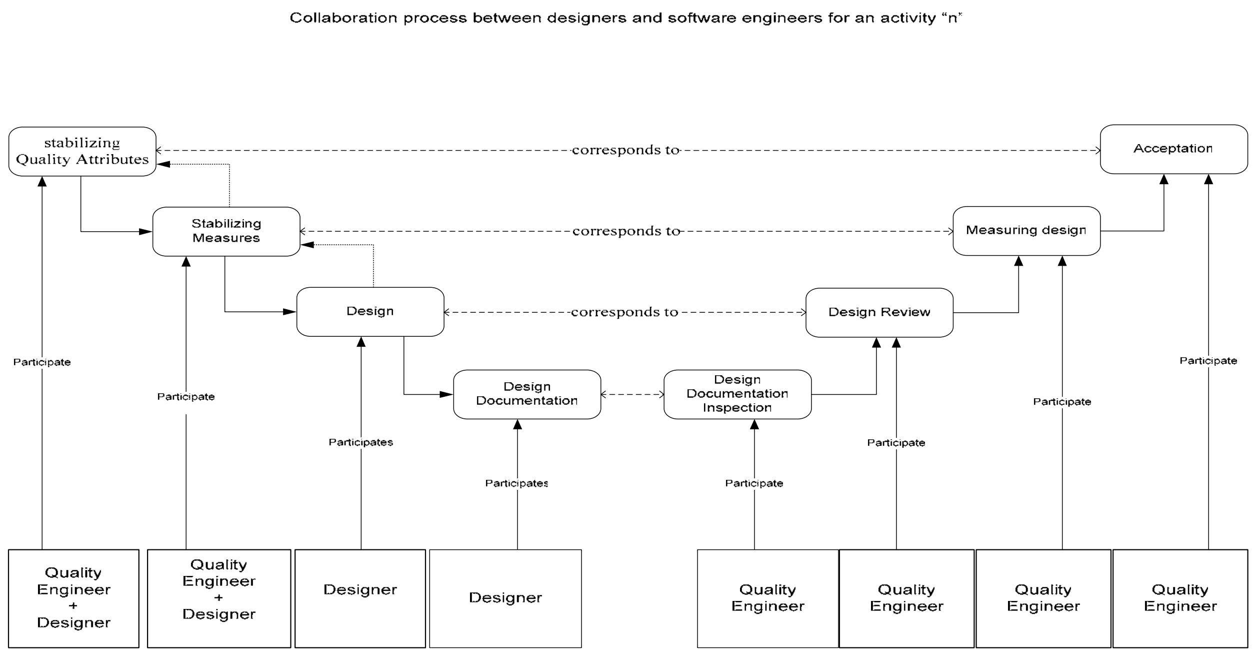


Figure 4: Collaboration process between designers and software engineers

7. Conclusion

In this paper we identified a gap in previous product quality models with respect to Software Under Development. An extension of the Suryn-Abran model to include components and measures for the phase of Conceptual Design which encapsulates two rather critical components namely *System Design* and *Program Design.* The model emphasised the need to integrate the development process with the Software Quality Engineering process as a preventative measure rather than relegating quality assurance activities to the end of the process,

Further work needs to be carried out in order to adjust and refine existing measures as well as propose measurement programmes and further measures that satisfy the most critical phases in software development, especially architecture and program design phases.

Acknowledgments

The authors express sincere appreciation to Mr. Jean-Francois Levesque, master student at Software and Information Technology Engineering Dept, École de technologie supérieure for his valuable contribution in the research program presented in this paper.

8. Bibliography

1. Suryn W., Abran A., Laporte C.Y., "An integrated life cycle quality model for general public market software products". Proceedings of 12th International Software Quality Management & INSPIRE Conference (BSI) 2004, Canterbury, Kent, UK 5-7 April 2004
2. ISO/IEC 9126 – Software Engineering – Product quality – Part 1: Quality model, 1999
3. ISO/IEC 9126 – Software Engineering – Product Quality – Part 3 – Internal Metrics, 2001
4. IEEE 1016 – Recommended Practice for Software Design Descriptions, 1998
5. ISO/IEC 14598 – Software Product Evaluation – Part 2: Planning and Management, 1995
6. ISO/IEC 14598 – Information technology – Software product evaluation Part 3: Process for developers, 1996
7. ISO/IEC 14598 – Software Engineering – Product evaluation – Part 6: Documentation of evaluation modules, 1999
8. IEEE 1058 – IEEE Standard for Software Project Management Plans, 1998
9. Roger S. Pressman, "*Software Engineering: A practitioner's approach, fourth edition*", McGraw-Hill Series, 1997
10. David Budgen, "*Software Design, second edition*", Pearson, 2003

11. Dick Hamlet, Joe Maybee, “*The Engineering of Software*”, Addison Wesley Longman Inc, 2001.
12. Georgiadou Elli " Software Process and Product Improvement A Historical Perspective", International Journal of Cybernetics, Volume 1, No1, Jan 2003 pp172-197